Praktikum
Entwicklung von Kommunikationssystemen (SoSe 2000)

Prof. Reinhard Gotzhein, Philipp Schaible

Aufgabe 1

Nachdem Ihr SDT kennengelernt habt und auch SDL in Erinnerung gebracht wurde, wollen wir mit der Entwicklung der Entwurfspezifikation unseres Kommunikationssystems beginnen. Wie es bereits in der Einführung erwähnt wurde, wollen wir die Spezifikation schrittweise entwikkeln. In diesem ersten Schritt entwickeln wir eine Spezifikation, welche folgender Problemstellung entspricht.

- Es sollen zunächst nur drei Anwendungen (drei Benutzer) existieren.
- Es wird davon ausgegangen, daß das Anmelden erfolgreich abgeschlossen ist.
- Einen öffentlichen Chat-Kanal öffnen und schließen.
- Text-Daten (Charstring) über diesen Kanal übertragen.
- Die Kommunikation zwischen den Protokollinstanzen wird als zuverlässig angesehen.
  Bemerkung: zuverlässige Übertragung kann durch einen SDL-Kanal modelliert werden.

Die Formulierung der entsprechenden Anforderungen im Detail kann dem Anforderungsdokument entnommen werden.

a) Erstellt ein SDL-Package[1] ***StepSigDat*** (**Sig**nal -und ***Dat***entypen für den ersten Schritt), wo die in diesem Schritt notwendigen Signale und Datentypen definiert sind. Dieses Package wird später in das Kommunikationssystem eingebunden. Syntaxüberprüfung kann auch für ein einzelnes Package mit dem Analyser erledigt werden.

   Bemerkung: Damit das System leichter veränderbar ist, sollte man für die Parameter spezielle Datentypen definieren. Wenn das im Moment kein komplexer Datentyp ist, kann man diesen mit SYNTYPE auf einfache SDL-Datentypen wie Integer oder Charstring abbilden. Z.B.

   SYNTYPE reasonType = Charstring ENDSYNTYPE;
   ...
   DCL reason reasonType;

   Bemerkung: Bei der Definition von Datentypen und auch generell für alle Textelemente der Spezifikation ist **<Window> <Grammar Help>** sehr nützlich. Zusammen mit Grammatikregeln befinden sich dort auch viele Beispiele für oft gebrauchte Sprachkonstrukte.

1. Definition eines Packages hat den Vorteil, die Signal -und Datentypen zusammenzuhalten und dabei an mehreren Stellen verwenden zu können.

b) Erstellt eine Spezifikation für das Kommunikationssystem mit drei Clients *client1*, *client2* und *client3* und einem Server *server* welche über SDL-Kanäle verbunden sind und mit den Anwendungen *appl1*, *appl2* und *appl3* entsprechend kommunizieren.

> **Bemerkung1:** Die Anwendungen selbst sind kein Teil des Kommunikationssystems und werden deswegen nicht mitspezifiziert. Nichtsdestotrotz müssen wir die Schnittstelle zu diesen Anwendungen spezifizieren. Hier macht man sich die Möglichkeit zu nutze, offene[1] Systeme in SDL zu spezifizieren.
>
> **Bemerkung2**: Verwendet BlockTypes und ProcessTypes anstatt Prozesse und Blöcke direkt zu spezifizieren[2]. Alle diese Typedefinitionen sollen in einem weiteren Package "***StepClasses***" zusammengefaßt werden. Dieser und das *StepSigDat* Package werden mit *USE "PackageName"* in das System eingebunden.

c) Übersetzt die Spezifikation mit dem Analyser. Benutzt dabei die <Generate><Make><Code generator>**<C_Advanced>** Einstellung. Diese Einstellung sollte man auch für alle weiteren Aufgaben beibehalten.

d) führt Simulationen durch. (Es geht primär darum, die durch den Simulator angebotenen Features selbst auszuprobieren. Es kann aber auch sein, daß man im Laufe der Simulation Fehler entdeckt.)

   i) Führt die Simulation Schritt für Schritt durch. (Durch [Trace] [SDL] kann man den Simulationsverlauf in den SDL Diagrammen beobachten.) Schaut Euch bei der Simulation die Inhalte der Prozeßwarteschlangen, Variablen, ReadyQueue und alles was euch sonst noch interessiert. Auch <Watch Window> und <Command Window> sollte man ausprobieren.

   > **Bemerkung**: Für das Verschicken der Umgebungssignale (in diesem ersten Schritt ist es nur *DATA*) definiert man sich besser eigene Knöpfe, so daß das Signal "per Knopfdruck" verschickt werden kann. (Siehe online-Help)

---

1. Systeme, die mit ihrer Umgebung kommunizieren können.
2. Dies erleichtert die Wiederverwendung der Ergebnisse dieses Schrittes.

ii) Führt die Simulation erneut durch. Diesmal versucht aber das Geschehen zu beeinflussen. Z.B. Kann man ein Signal aus einer Warteschlange entfernen. Auch die Reihenfolge der Transitionsausführung läßt sich verändern. Denkt euch ein Szenario aus, bei dem Ihr dies ausprobieren könnt.

iii) Der Verlauf der Simulation läßt sich auch über Breakpoints verfolgen. Dabei ist es möglich, Breakpoints an Transitionen, Signalen und einzelnen SDL-Symbolen anzubringen. Probiert alle drei dieser Möglichkeiten aus. Die Signal- und Transition-Breakpoints werden direkt aus dem Simulator über ein Menü mit Fragen und Antworten gesetzt. (Kleiner Tip: nicht alle Angaben müssen gemacht werden; Im Zweifelsfall einfach zur nächsten Frage mit <RETURN> übergehen. Die Symbol-Transitionen setzt man am besten direkt im SDL Diagramm. Vorher muß man allerdings die SDL-Diagramme an den Simulator anbinden <Breakpoint> <Connect sdle>. Danach einfach das gewünschte Symbol markieren und <Breakpoints><Set Breakpoint> drücken.